2021-07

SCRAMBLE: Sweep Comparison Research Application for Multiple Back-Gated FieLd Effect measurements of graphene field effect transistors

O'Driscoll, B

10.1016/j.softx.2021.100757
SoftwareX
Elsevier

SoftwareX 15 (2021) 100757

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

SCRAMBLE: Sweep Comparison Research Application for Multiple Back-Gated FieLd Effect measurements of graphene field effect transistors

Benjamin O'Driscoll

Wolfson Nanomaterials and Devices Laboratory, School of Engineering, Computing and Mathematics, University of Plymouth, Devon, PL4 8AA, UK

ARTICLE INFO

Article history:
Received 13 October 2020
Received in revised form 16 June 2021
Accepted 29 June 2021

Keywords:
Graphene field effect transistors (GFETs)
Dirac points
Field effect mobility
Maximum transconductance

ABSTRACT

Graphene Field Effect Transistors (GFETs) are active electronic components which exploit the modulation of charge carriers in a graphene channel for a wide variety of applications such as electronic switching, amplification and biosensing. Herein describes a new software package SCRAMBLE, written in Python and developed with a graphical user interface, to speed up the routine processing of multiple back-gated electrical measurements from GFETs. For forward and reverse sweeps, SCRAMBLE automatically determines the Dirac points, positions of maximum transconductance and calculates the field effect mobilities for electrons and holes.

Code metadata

Current code version	*v1.0*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-20-00068
Code Ocean compute capsule	*N/A*
Legal Code Licence	*MIT*
Code versioning system used	*git*
Software code languages, tools, and services used	*Python.*
Compilation requirements, operating environments & dependencies	*Python modules requiring installation:* 1. *Pandas* 2. *NumPy* 3. *Matplotlib 3.2.2* 4. *tkinter*
If available Link to developer documentation/manual	https://github.com/phukeo/SCRAMBLE/blob/master/README.md
Support email for questions	benjamin.odriscoll@plymouth.ac.uk;

1. Motivation and significance

Graphene field effect transistors (GFETs) are realised by connecting a graphene channel between metallic source and drain electrodes, operating in either a top- or back-gated configuration. Current is passed through the conducting graphene channel when a voltage drop is created across the device electrodes. Modulating the charge density in the channel of GFETs is achieved by the application of an external voltage to the gate terminal inducing carriers by field-effect [1] (Fig. 1A).

The fundamental electrical characterisation for GFETs is the $I_{SD} - V_G$ gated sweep. During this measurement, the source–drain voltage (V_{SD}) is maintained constant and the current (I_{SD}) through the graphene channel is recorded whilst the gate terminal voltage (V_G) is swept forward from one value V_{GMin} to V_{GMax} and then back again from V_{GMax} to V_{GMin} (Fig. 1B). During this $I_{SD} - V_G$ gated sweep measurement, charge carriers in the graphene channel are modulated by the applied electric field, changing the conductive properties of the GFET [1]. The positions of minimum conductance, referred to as Dirac points (V_{DF} and V_{DR} in Fig. 1B) correspond to locations where the Fermi level is at the intersection between the conduction and valence bands, which describes where the availability of the electron and hole charge carriers respectively are at their minimum [2]. Analysis of the Dirac points and transfer curve characteristics provides information on the doping and mobility of the GFET under test [2,3]. Additional information related to the graphene surface and interface charges can be obtained by comparing the forward and

E-mail address: benjamin.odriscoll@plymouth.ac.uk.

https://doi.org/10.1016/j.softx.2021.100757

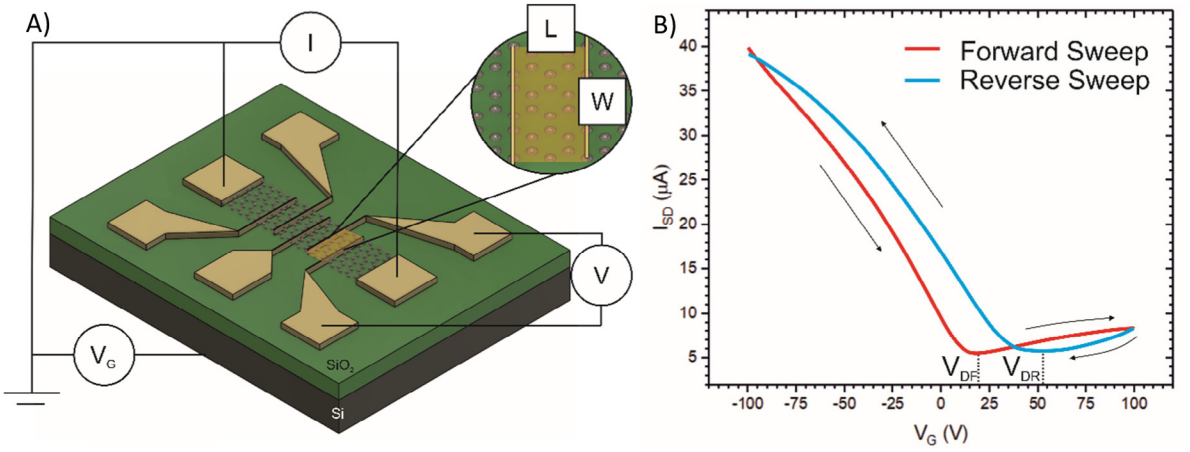

Fig. 1. (A) Typical GFET schematic showing experimental setup for an $I_{SD} - V_G$ measurement, (B) Representative $I_{SD} - V_G$ data showing the difference in the Dirac points for forward and reverse sweeps.

reverse sweep characteristics as hysteretic behaviour is observed (Fig. 1B) [4].

Large amounts of information can be elucidated from a single $I_{SD} - V_G$ sweep measurement relating to a host of sensing characteristics related to the device under test. Usually, multiple repeats under different environmental conditions, with different functionalisation regimes are conducted resulting in significant amounts of data to process, analyse and visualise.

SCRAMBLE has been developed to provide a platform with an easy-to-use graphical user interface (GUI) that allows researchers of all software skill levels to swiftly and accurately monitor key parameters relating to GFETs in large datasets, allowing them to make data driven conclusions relating to their devices. Since GFET devices are used across a plethora of different research areas including medical diagnostics [5,6], food adulteration testing [7,8] and novel electronic component design such as logic gates [9] and logic inverters [10], SCRAMBLE could have a substantial impact across a wide range of exciting fields of study. This software package will support researchers working in the field of GFETs by enabling them to rapidly and accurately process datasets. SCRAMBLE automates all aspects of data manipulation, transforming raw data from measurement equipment into useful visualisations, whereby complex parameters (such as Dirac points and mobility values) are calculated and determined at the click of a single button, thus eliminating human error. In addition, as SCRAMBLE's source code remains open and adaptable, users with different characterisation procedures and equipment can easily customise the software to their exact purposes.

2. Software description

The main components of SCRAMBLE's GUI are shown in Fig. 2. The front panel is split into the left hand side control panel and the visualisation screen. The control panel contains several inputs and controls that allows the user to import raw .csv files, customise the dimensions and parameters of their devices, select $I_{SD} - V_G$ sweep(s) of choice, average, plot and also export data. The visualisation screen displays the interactive charts once the "Process Data" button is pressed.

2.1. Software architecture

The source code for SCRAMBLE is written in Python, and is split across two modules, "scrambleGUI" and "scrambleFUN" as depicted in Fig. 3. The "scrambleGUI" module handles all aspects of the GUI including buttons, inputs and charts by implementing code from the tKinter library [11]. The "scrambleFUN" module is where the computation of parameters is performed. Here, data is imported, manipulated and analysed according to specific functions. Other open source libraries involved in this software package include os, NumPy [12], Pandas [13] and Matplotlib [14].

The user begins by loading in the raw .csv files from folders of their choice corresponding to the multiple $I_{SD} - V_G$ data acquisitions using the "Open Folder" button within the "Data Selection" frame. An algorithm selects only the two data columns of interest (I_{SD} vs. V_G) and ignores the metadata situated in the header rows. The names of the imported files are printed in the list-box within the GUI and the data is held in memory. Note that the "Open. BOD" button should be used on previously exported data to prevent unnecessary computational effort, which reduces re-import times for large datasets.

Next, the user must update the inputs of the "Device Parameters" frame (Fig. 2) according to the details/dimensions of their GFETs. Here, "A" is the source–drain voltage V_{SD} in mV, "B" is the device length in μm, "C" is the device width in μm (Fig. 1A), "D" is the dielectric thickness t_{ox} in nm and "E" is the dielectric constant ε_r in Fm^{-1}.

The user then decides which of the imported $I_{SD} - V_G$ sweeps they wish to evaluate, by selecting one or more of the labelled names in the "Data List" frame. Next, the user must decide whether they want the data visualised as either current or resistance against V_G. If the user selects the "Resistance" option then both the resistance in ohms (Ω) and the sheet resistance in Ohms/square (Ω/□) will be displayed in the "Sweep Visualisation" frame, on the left hand and right hand y-axis respectively. The sheet resistance of 2D materials is described in more detail here [15]. Note that for the remainder of this contribution it is assumed that the user has selected the "Current" option.

To begin the core algorithm the user then presses the "Process Data" button, which commences the workflow depicted in Fig. 3 to determine and calculate the characterisation parameters. For each of the $I_{SD} - V_G$ sweeps selected the following parameters are determined for both the forward and reverse sweeps; current/voltage values of Dirac Points, current value at $V_G = 0$, current/voltage values of maximum transconductance points and the electron and hole mobilities.

For each of these parameters, the forward and reverse values are plotted on the same chart with the difference between them highlighted by a colour coded line which determines whether or not the forward (black) or reverse (red) occurred at a greater value. Positions corresponding to the Dirac points and maximum transconductance are overlaid onto the "Sweep Visualisation" chart (Fig. 2).

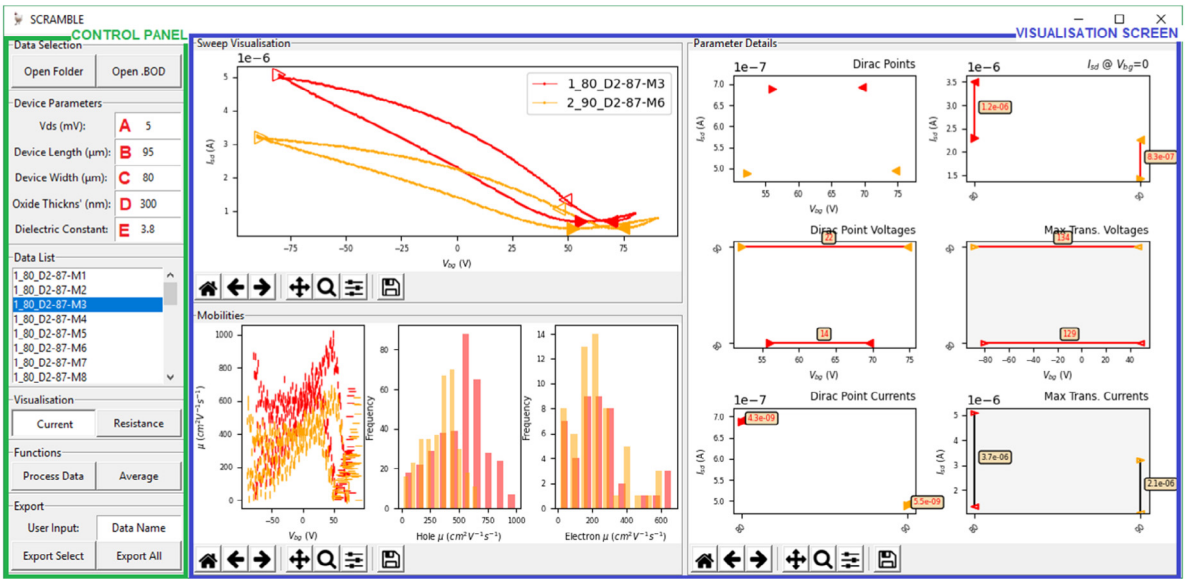

Fig. 2. SCRAMBLE GUI showing the control panel with input parameters and visualisation screen where data is represented in various charts. A comparison between sweeping V_G between $-/+$ 80 V (red) and $-/+$ 90 V (yellow) is shown. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

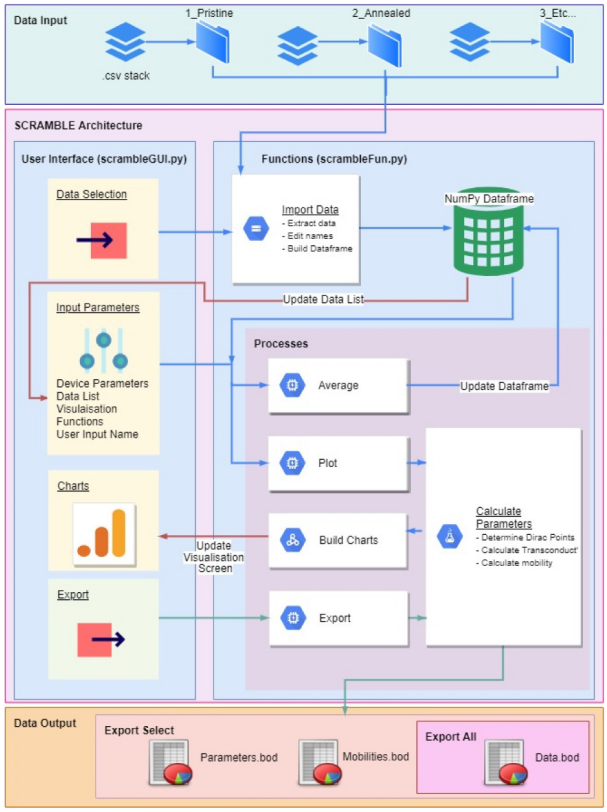

Fig. 3. SCRAMBLE software architecture overview showing two modular components to the package along with the data input and output locations. Raw data is pulled into the scrambleFUN.py module where it is manipulated based on the input parameters from the control panel handled by the scrambleGUI.py module. A NumPy DataFrame is constructed and updates the "Data List" for the user to choose which measurements they wish to visualise. The processing pathways for averaging, plotting and exporting data are shown with the appropriate arrows. The file types exported by SCRAMBLE are shown under the appropriate button name.

SCRAMBLE's source code has been left fully adaptable so that the customisation of data import from different Semiconductor Device Analysers (SDAs) and templates along with defining user specific default values for GFET details/dimensions is easily achieved (Listing 1). Further details on the customisation and practical use of SCRAMBLE are given in the online README.md.

2.2. Software functionalities

As the $I_{SD} - V_G$ sweeps consist of a forward and reverse direction, the first part of the data manipulation involves splitting the data into these directions. Then, by selecting the minimum current value and its corresponding index for the back-gate voltage, the position of the Dirac points are determined.

The transconductance ($g_m = \partial I_{SD}/\partial V_G$) is computed using the "gradient" function from the "NumPy" Python module [12]. The maximum transconductance values and their corresponding positions in current and voltage space are then determined.

The mobility is described by hole and electron conduction, referring to which charge carrier is the majority contributor to the current through the graphene channel at a particular V_G. At the Dirac point, the voltage of minimum conductance, the Fermi level sits at the intercept between the valence (majority hole) and conduction (majority electron) bands. Conduction in regions below (above) the Dirac point voltage show majority hole (electron) charge carrier contributions [2]. Therefore, once the Dirac points have been identified, they are used to split the raw data into the regions which describe hole and electron conduction.

The field effect mobility is calculated using the direct transconductance method (DTM) described in more detail in [16]. This method relates the mobility of the device with the transconductance using Eq. (1).

$$\mu_{DTM} = g_m \frac{L}{W V_{SD} C_G} \tag{1}$$

where μ_{DTM} is the field effect mobility, g_m is the transconductance, L, W are the length and width of the device respectively, V_{SD} is the source–drain voltage with $C_G = \varepsilon_r \varepsilon_0 / t_{ox}$ describing the back-gate capacitance, where ε_r is the relative permittivity, ε_0 is the permittivity of free space and t_{ox} is the thickness of the insulating layer [16].

```
sourceDrainLabel=tk.Label(framel, text="Vds (mV):",padx=5, pady=5, width=w)
sourceDrainLabel.grid(row=0,column=0,sticky="nesw")
sourceDrainEntry=tk.Entry(framel, justify="center", width=w)
sourceDrainEntry.grid(row=0, column=1,sticky="nesw")
sourceDrainEntry.insert(0,5)
deviceLengthLabel=tk.Label(framel, text="Device Length (\u03bcm):",padx=5, pady=5, width=w)
deviceLengthLabel.grid(row=1, column=0,sticky="nesw",columnspan=1)
deviceLengthEntry=tk.Entry(framel, justify="center", width=w)
deviceLengthEntry.grid(row=1, column=1,sticky="nesw",columnspan=1)
deviceLengthEntry.insert(0,95)
deviceWidthLabel=tk.Label(framel, text="Device Width (\u03bcm): ",padx=5, pady=5, width=w)
deviceWidthLabel.grid(row=2, column=0,sticky="nesw",columnspan=1)
deviceWidthEntry=tk.Entry(framel, justify="center", width=w)
deviceWidthEntry.grid(row=2, column=1,sticky="nesw",columnspan=1)
deviceWidthEntry.insert(0,80)
oxideThickLabel=tk.Label(framel, text="Oxide Thickns' (nm): ",padx=5, pady=5, width=w)
oxideThickLabel.grid(row=3, column=0,sticky="nesw",columnspan=1)
oxideThickEntry=tk.Entry(framel, justify="center", width=w)
oxideThickEntry.grid(row=3, column=1,sticky="nesw",columnspan=1)
oxideThickEntry.insert(0,300)
oxideDielecLabel=tk.Label(framel, text="Dielectric Constant: ",padx=5, pady=5, width=w)
oxideDielecLabel.grid(row=4, column=0,sticky="nesw",columnspan=1)
oxideDielecEntry=tk.Entry(framel, justify="center", width=w)
oxideDielecEntry.grid(row=4, column=1,sticky="nesw",columnspan=1)
oxideDielecEntry.insert(0,3.8)
```

Listing 1. Code snippet from scrambleGUI.py module showing lines of code associated with default values for the control panel which are easily customised by users to correspond with values associated with their own device dimensions and parameters.

DTM does not factor in contributions from the contact resistance and therefore always estimates a value lower than the real mobility. This method was chosen as its accuracy has shown to increase for larger channel sizes (>6 μm) which resemble the dimensions of the GFETs used in our own work [16].

The mobility data is plotted in three charts in SCRAMBLE. Firstly, the mobility at each point across the $I_{SD} - V_G$ sweep is plotted with the majority charge carrier at each point determined by the vertical or horizontal bars corresponding to hole and electron charge carriers respectively. Next, histograms detailing the frequency of binned mobility values for the hole and electron charge carriers are given.

SCRAMBLE offers the option to average multiple $I_{SD} - V_G$ sweeps together. This is achieved by firstly selecting the $I_{SD} - V_G$ sweeps from the "Data List" and then pressing the "Average" button. The newly created $I_{SD} - V_G$ sweep is appended to the list with the name formed of the entry in the "User Input" and "AVE" concatenated to the end to signify that it has been processed within SCRAMBLE. This can then be selected for evaluation as previously discussed.

The individual charts shown in Fig. 2 can be manipulated with their individual navigation toolbars allowing panning, zooming and the configuration of subplots. Pressing the "Save" icon allows the user to export the plots in various formats such as Portable Network Graphics (PNG), Scalable Vector Graphics (SVG) and Raw RGBA bitmap to name a few.

The raw data and calculated parameters for $I_{SD} - V_G$ sweeps can be exported from SCRAMBLE using the "Export Selected" button which performs this on the data highlighted in the "Data List". This action produces three text files (.bod); one with the raw $I_{SD} - V_G$ data, one with the calculated mobility values and one with the determined parameters. These are labelled accordingly by concatenating the input of "User Input" entry box with "Data", "Mobilities" and "Parameters" respectively. Note that using the "Export All" button exports only the raw $I_{SD} - V_G$ data for all data within the "Data List" – it does not export calculated results.

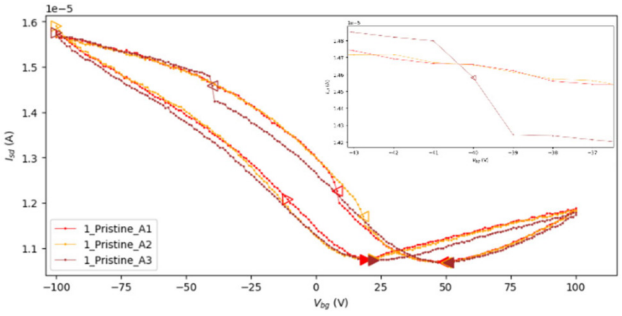

Fig. 4. Sweep Visualisation Plot for devices A1, A2 and A3, with inset showing an anomalous step in the current between −39 V and −41 V indicated by the left pointing unfilled triangle for A3, corresponding to the maximum transconductance for the reverse sweep.

3. Illustrative examples

The influence of applying conventional annealing at a temperature of 215 °C for 30 mins on a GFET device is demonstrated in this section to exhibit the main functionalities of SCRAMBLE. The fabrication details of the GFET used in this work can be found in [5]. In this work, three $I_{SD} - V_G$ sweeps were performed before (A1, A2 and A3) and after (B1, B2 and B3) the annealing process was conducted. The three raw $I_{SD} - V_G$ sweeps before the annealing treatment are shown in Fig. 4. In this plot, the positions of the forward and reverse (right and left) Dirac points and maximum transconductance points (filled and unfilled) are illustrated by suitable triangles. It is clear from the anomalous step clearly highlighted by SCRAMBLE at I_{SD} at −40 V that the A3 measurement should be excluded from future analysis.

Averaging and then plotting suitable values for before (A1 and A2) and after (B1, B2 and B3) the annealing process is completed next in order to evaluate the parameter change caused by the heating process. A summary of some of the salient charts are given in Fig. 5.

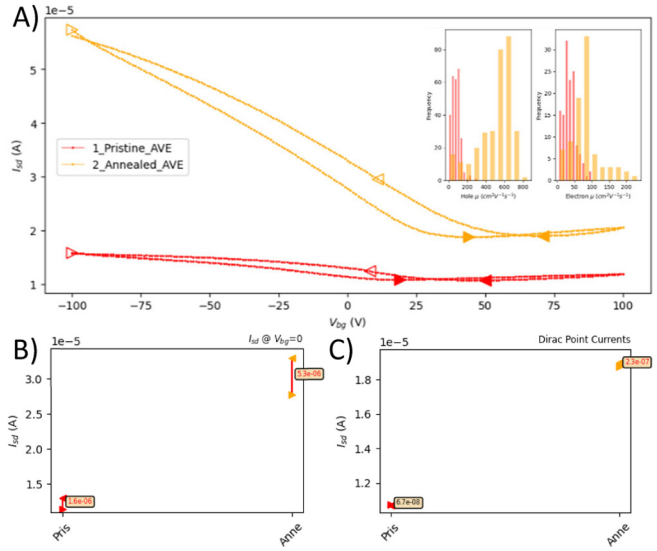

Fig. 5. (A) Visualisation screen showing the sweep characteristics for the pristine and annealed data with inset showing the corresponding mobility values. For the forward and reverse sweeps (B) shows the current values at $V_G = 0$ and (C) shows the current values for the Dirac points.

Using SCRAMBLE, it is swiftly shown that the annealing process has shifted the Dirac points to higher V_G and I_{SD} values, increased the current value through the GFET at all points and increased the hole (electron) mobilities in the graphene channel by three (two) times (Fig. 5A). The colour coded lines and text associated with the markers allow for swift conclusions to be made about how this process has influenced the relative positions of the forward and reverse sweeps (Fig. 5B and C). For example, the current values for V_{DF} and V_{DR} swap from being higher for V_{DF} at the pristine stage to being higher for V_{DR} at the annealing stage, indicated by the black and red text and bars respectively. All outputted data is given in the supplementary material.

Prior to SCRAMBLE's inception, the end-to-end processing of similar amounts of data would have taken several hours to conduct from collection to visualisation, with several hundred stages to manually complete offering several opportunities for the user to introduce error into the calculations. SCRAMBLE allows users to rapidly compare results maximising time elsewhere and has proved to be an essential tool for sharing critical results such as those shown in Fig. 5 with others in our group.

4. Impact

The primary impact of using SCRAMBLE on a daily basis has been the speed at which raw data is converted into informed decisions about next steps in an investigation. Specifically, key artefacts accurately calculated and clearly visualised by SCRAMBLE, that had previously been overlooked, have directed research into new routes of exploration for our group such as examining the effect of sweep velocity on GFET characteristics. SCRAMBLE eliminates the delay between acquiring, processing and visualising data as this is all completed in a few automated stages within the package. This means that valuable investigatory time can be applied to further the group's knowledge on how different functionalisation processes impact GFET properties or alternatively into improving experimental design. Data exported from SCRAMBLE is fully compatible with other software processing packages, such as Origin, SigmaPlot and Excel, which eases the transition from raw data to further data analysis.

SCRAMBLE is expected to contribute to any research group working on direct current measurements of GFETs. However, since Python is rapidly growing in popularity across many diverse fields of applications (AI/machine learning, Big Data etc.) it is envisaged that the software will be readily refined to suit a plethora of specific applications which involve field effect transistors, in general.

SCRAMBLE has been used in our research group by Ph.D. students for analysis purposes and also as a demonstration tool for Masters students whereby key parameters relating to GFET devices can be clearly visualised and explained. The software also enables students to cover more material during laboratory sessions, due to time saved from processing data, thereby assisting such students in gaining an enhanced understanding of the field. Its use as a training tool in this aspect facilitates the continuous progression of postgraduate students into the group which will secure the future of this research.

5. Conclusions

SCRAMBLE is software package which processes raw back-gated field effect sweep measurement data from GFETs into comparable metrics to support researchers to characterise devices in a reproducible manner. It is an easy to use platform, suitable for researchers with limited software skills, which provides key data insights relating to GFETs such as Dirac points and mobility calculations, visualised over several charts and output tables. The combination of SCRAMBLE's simple GUI along with its open and fully customisable source code will make this software package attractive to a large audience of researchers who are developing GFET technologies for a myriad of different applications thus facilitating the discovery of new and exciting avenues for future enquiry.

Declaration of competing interest

The author declares that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Thanks is given to Toby Whitley, Shakil Awan and David Jenkins for proof reading and useful discussions.

Funding is acknowledged from the University of Plymouth, UK GD110025-104.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2021.100757.

References

[1] Tran T-T, Mulchandani A. Carbon nanotubes and graphene nano field-effect transistor-based biosensors. TRAC Trends Anal Chem 2016;79:222–32.
[2] Reddy D, et al. Graphene field-effect transistors. J Phys D: Appl Phys 2011;45(1).
[3] Yang Y, Brenner K, Murali R. The influence of atmosphere on electrical transport in graphene. Carbon 2012;50(5):1727–33.
[4] Yang J, et al. Hysteresis analysis of graphene transistor under repeated test and gate voltage stress. J Semicond 2014;35(9):094003.
[5] Haslam C, et al. Label-free sensors based on graphene field-effect transistors for the detection of human chorionic gonadotropin cancer risk biomarker. Diagnostics (Basel) 2018;8(1):5.
[6] Kwong Hong Tsang D, et al. Chemically functionalised graphene FET biosensor for the label-free sensing of exosomes. Sci Rep 2019;9(1):13946.
[7] Chee LH, et al. DNA/AuNP-graphene back-gated field effect transistor as a biosensor for lead (II) ion detection. In: 2017 IEEE regional symposium on micro and nanoelectronics (RSM). 2017.

[8] Kim D, et al. Electrical conductance change of graphene-based devices upon surface modification for detecting botulinum neurotoxin. Japan J Appl Phys 2017;56(6):067001.
[9] Sordan R, Traversi F, Russo V. Logic gates with a single graphene transistor. Appl Phys Lett 2009;94(7):073305.
[10] Kim E, et al. Logic inverter implemented with CVD-assembled graphene FET on hexagonal boron nitride. IEEE Trans Nanotechnol 2012;11(3):619–23.
[11] Lundh F. An introduction to tkinter. 1999, URL: http://www.jgaltier.free.fr/Terminale_S/ISN/TclTk_Introduction_To_Tkinter.pdf. Accessed 06/07/2021.
[12] Harris CR, et al. Array programming with numpy. In: A guide to NumPy. Vol. 1. Nature 2020;585(7825):357–62.
[13] McKinney W. Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science Conference,. 2010, p. 56–61. http://dx.doi.org/10.25080/Majora-92bf1922-00a.
[14] Hunter JD. Matplotlib: A 2D graphics environment. Comput Sci Eng 2007;9(3):90–5. http://dx.doi.org/10.1109/MCSE.2007.55.
[15] Bøggild P, et al. Mapping the electrical properties of large-area graphene. 2D Mater 2017;4(4):042003.
[16] Zhong H, et al. Comparison of mobility extraction methods based on field-effect measurements for graphene. AIP Adv 2015;5(5):057136, licensed under a Creative Commons Attribution (CC BY) licence.